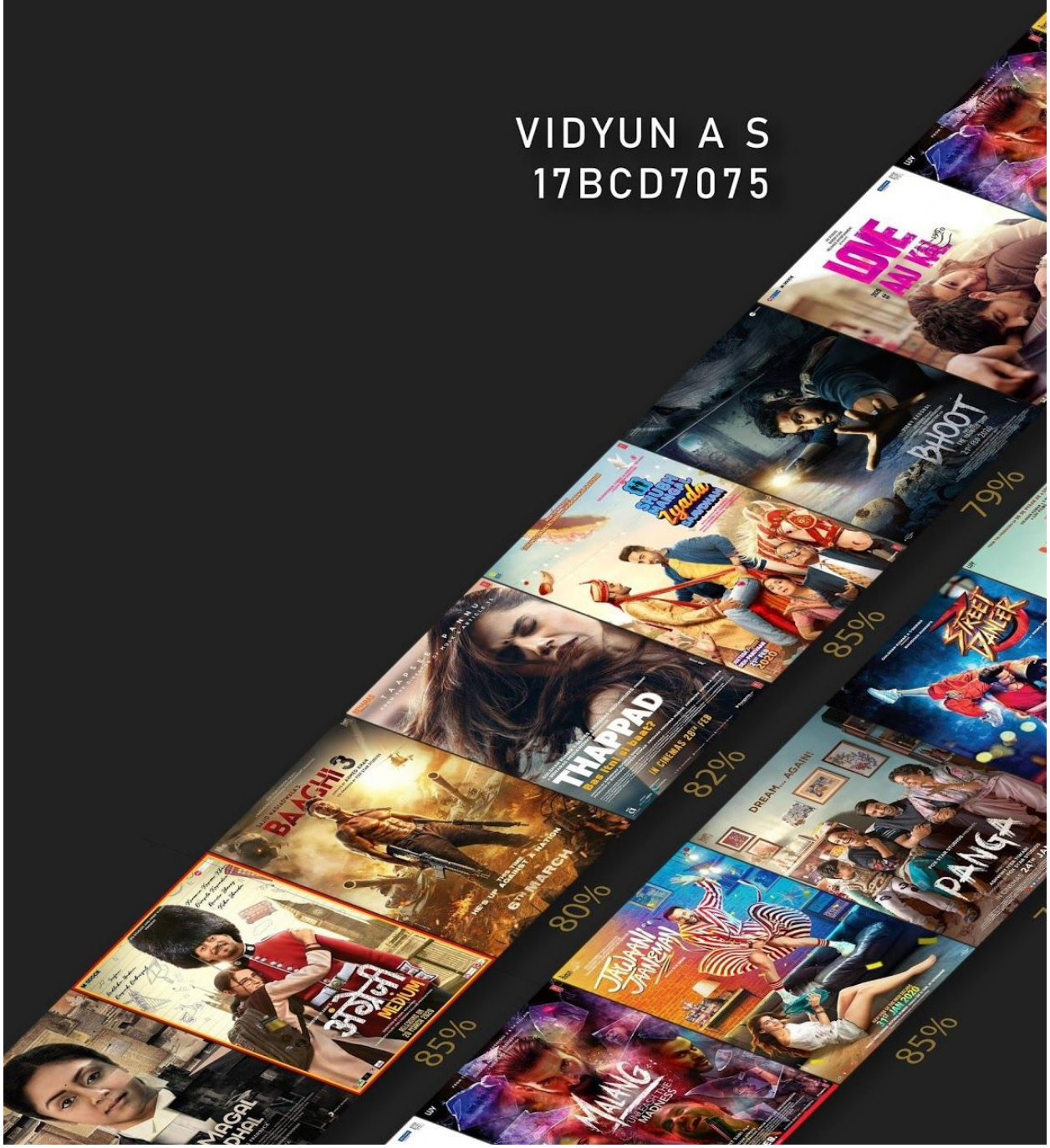
MOVIE REVIEW SYSTEM USING FLASK

VIDYUN A S
17BCD7075



ACKNOWLEDGEMENT

On the very outset of this project, I would like to extend my heartfelt obligation to all the people who helped me the most throughout the completion of the project.

I am extremely thankful and pay my gratitude to my faculty, Prof. Hari Kishan for his guidance and support for my project.

I wish to thank my parents and family for their unconditional love and support which inspired me to go my own way.

Special thanks to my colleagues who helped me out in completing the project, by exchanging their interesting ideas, thoughts and made it possible to complete my project with all accurate information.

Last but not the least, I want to thank my friends who treasured me for my hard work and encouraged me and above all, I would like to thank the Almighty for being able to complete this project with success.

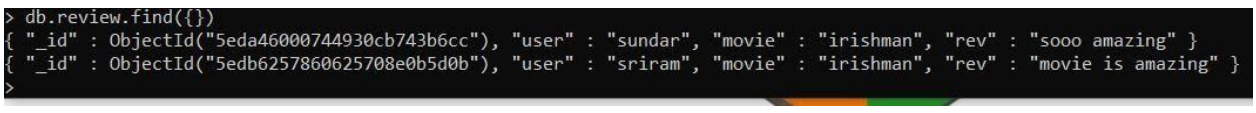
ABSTRACT:

The craze for watching movies will never decrease and during the current situation, the increase of people watching movies on internet platforms is burgeoning. It is necessary for Multinational Companies to invest in the best movies and make them available to the proper target audience. Using the Flask app which is proposed, this can be achieved and will be helpful to find the target audience and companies to invest on propitious movies. In this app, people login or create an account and provide reviews about the movie. These reviews are stored in MONGODB database and Sentiment Analysis and scores are provided respectively. Finally, one can check the review of the movie, where it displays the average audience score with their gender information and reach of the movie in different states of the country.

TABLE OF CONTENT:

DATABASE:

The project uses MONGO database with two collections, “users” and “reviews” where, User collection stores username, password, location, sex of the user and reviews with the store review, username, movie name. Using the username as the similar value between these collections, the data between two collections are analyzed.



HTML PAGES:

There are a total of 5 html pages created-login, signup, main page, check review and give review with different functionalities according to the name. The output section will show you the output and page structure.

CODE:

Since most of the html codes are well known, i am adding only the python backend code here, other documents are attached along with this report.

```
from flask import Flask,redirect,url_for,render_template,request,session
from pymongo import MongoClient
client = MongoClient("mongodb://localhost:27017/")
mydatabase = client["projdb"]
mycollection = mydatabase["users"]
import pandas
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
nltk.download('vader_lexicon')
import statistics
import matplotlib.pyplot as plt
import collections
import os



app=Flask(__name__)
app.config['SEND_FILE_MAX_AGE_DEFAULT'] = 1
@app.route("/",methods=["POST","GET"])
def home():
        if request.method=="POST":

                user=request.form["uname"]
                pas=request.form["psw"]
                result=mycollection.find_one({"name":user,"pw":pas})
                if(result):
                        session["user"]=user
                        return render_template("chumma.html")
                else:
                        return render_template("login.html")
```

```
        else:
                return render_template("login.html")


@app.route("/signup",methods=["POST","GET"])
def sign():
        if request.method=="POST":
                user=request.form["name"]
                pas=request.form["psw"]
                rpas=request.form["rpas"]
                state=request.form["state"]
                sex=request.form["sex"]
                if(pas==rpas):
                        rec={ "name":user, "pw":pas, "location":state, "sex":sex}
                        x= mydatabase.users.insert_one(rec)
                        if(x):
                                return render_template("login.html")
                else:
                        return "password and confirm password not same"

        else:
                return render_template("signup.html")


@app.route("/main",methods=["POST","GET"])
def mainpage():
        if request.method=="POST":
                a=request.form["button"]
                lis=a.split()
                movie=lis[0]
                act=lis[1]
                session["moviename"]=movie
                if(act=="give"):

                        return render_template("give_review.html",moviename=movie)

                else:
                        return redirect(url_for("see"))
```

```
        else:
                return render_template("chumma.html")

@app.route("/give",methods=["POST","GET"])
def give():
        mycollection = mydatabase["review"]
        if request.method=="POST":
                review=request.form["review"]

result=mycollection.find_one({"user":session["user"],"movie":session["moviename"]})
                if(result):
                        return "<h1>you already gave the review</h1> "


                else:
                        rec={"user":session["user"], "movie":session["moviename"], "rev":review}
                        x= mydatabase.review.insert_one(rec)
                        if(x):
                                return render_template("chumma.html")

        else:
                return render_template("give_review.html")

@app.route("/see")
def see():


        mycollection = mydatabase["review"]
        mycollection2 = mydatabase["users"]
        moviename=session["moviename"]
        obj = SentimentIntensityAnalyzer()

        reviews=[]
        username=[]
        value=[]
        sex=[]
        loc=[]
        cursor = mycollection.find({"movie":moviename})
        for record in cursor:
                reviews.append(record["rev"])
                username.append(record["user"])
```

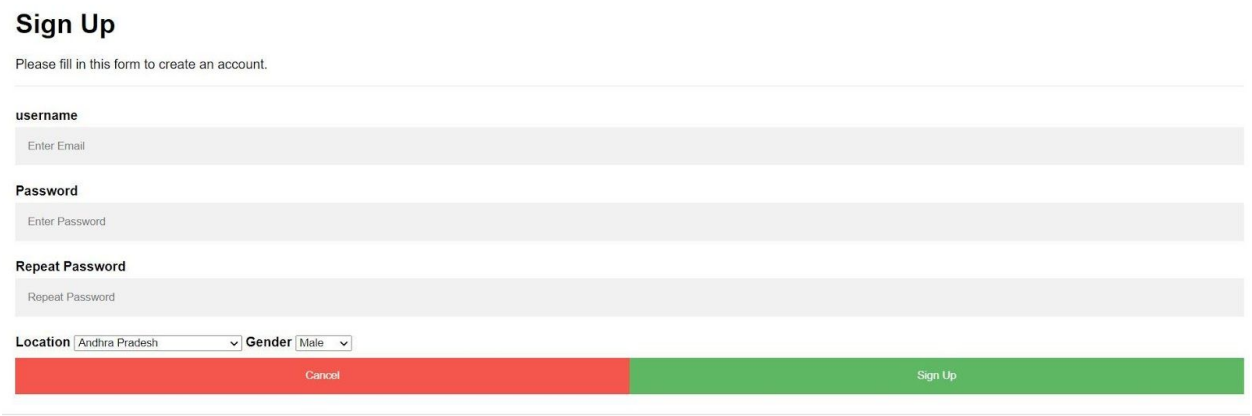
```
        for i in reviews:
                sentiment_dict = obj.polarity_scores(i)
                value.append(50*(sentiment_dict['compound']+1))
        for i in username:
                cursor=mycollection2.find({'name':i})
                for temp in cursor:
                        sex.append(temp["sex"])
                        loc.append(temp["location"])


        freq_loc = collections.Counter(loc)
        freq_sex = collections.Counter(sex)
        myset = set(loc)
        unique_loc = list(myset)
        myset = set(sex)
        unique_sex = list(myset)
        unique_count_loc=[]
        unique_count_sex=[]
        for i in unique_loc:
                unique_count_loc.append(freq_loc[i])
        for i in unique_sex:
                unique_count_sex.append(freq_sex[i])



        plt.pie(unique_count_sex, labels = unique_sex, shadow = True,startangle =
90,autopct='%1.2f%%')
        plt.title('reach of the movie among gender')

        plt.tight_layout()
        plt.savefig('C:/Users/vidyu/Desktop/flask_project_final/static/images/a.png')



        plt.pie(unique_count_loc, labels = unique_loc, shadow = True,startangle =
90,autopct='%1.2f%%')
        plt.title('reach of the movie among states')
        plt.tight_layout()
        plt.savefig('C:/Users/vidyu/Desktop/flask_project_final/static/images/b.png')
```

```
        return render_template("see_review.html",average=str(statistics.mean(value)))


@app.after_request
def add_header(response):

    response.headers['X-UA-Compatible'] = 'IE=Edge,chrome=1'
    response.headers['Cache-Control'] = 'public, max-age=0'
    return response

@app.after_request
def add_header(response):
    # response.cache_control.no_store = True
    if 'Cache-Control' not in response.headers:
        response.headers['Cache-Control'] = 'no-store'
    return response









if __name__=="__main__":
        app.secret_key="it is radom"
        app.run()
```

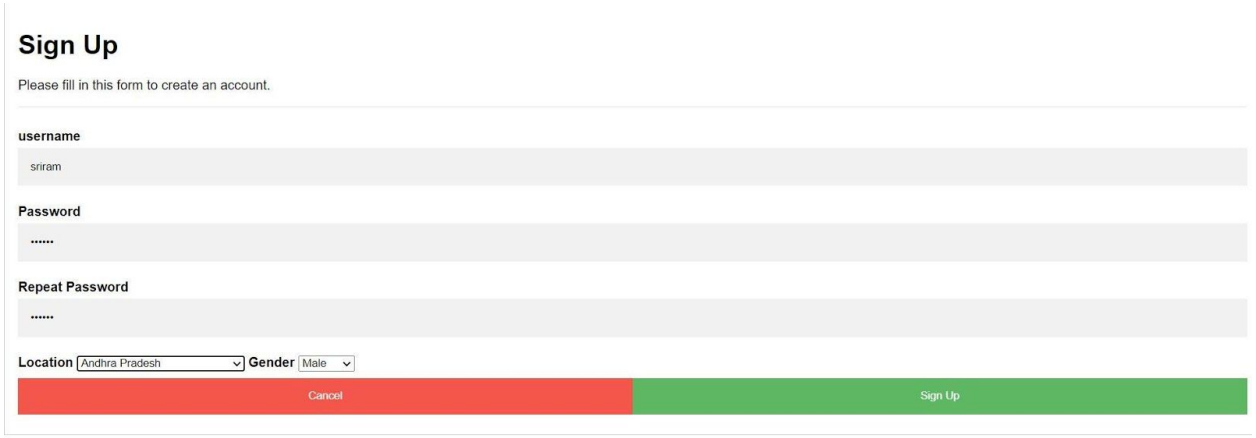
OUTPUT :

INITIALLY APP STARTS WITH THIS PAGE:

THEN YOU CAN CREATE AN ACCOUNT USING THE SIGN UP BUTTON WHICH REDIRECTS YOU:



ONCE YOU PROVIDE THE INPUT, THE DATABASE IS UPDATED:




```
> use projdb
switched to db projdb
> db.users.find({})
{ "_id" : ObjectId("5eda45e50744930cb743b6cb"), "name" : "sundar", "pw" : "sundar", "location" : "Tamil Nadu", "sex" : "
male" }
> db.users.find({})
{ "_id" : ObjectId("5eda45e50744930cb743b6cb"), "name" : "sundar", "pw" : "sundar", "location" : "Tamil Nadu", "sex" : "
male" }
{ "_id" : ObjectId("5edb608c860625708e0b5d08"), "name" : "vidyun", "pw" : "vidyun", "location" : "Delhi", "sex" : "male"
 }
{ "_id" : ObjectId("5edb60a1860625708e0b5d09"), "name" : "devi", "pw" : "devi", "location" : "Tamil Nadu", "sex" : "fema
le" }
>
```

```
> db.users.find({})
{ "_id" : ObjectId("5eda45e50744930cb743b6cb"), "name" : "sundar", "pw" : "sundar", "location" : "Tamil Nadu", "sex" : "
male" }
{ "_id" : ObjectId("5edb608c860625708e0b5d08"), "name" : "vidyun", "pw" : "vidyun", "location" : "Delhi", "sex" : "male"
 }
{ "_id" : ObjectId("5edb60a1860625708e0b5d09"), "name" : "devi", "pw" : "devi", "location" : "Tamil Nadu", "sex" : "fema
le" }
{ "_id" : ObjectId("5edb617a860625708e0b5d0a"), "name" : "sriram", "pw" : "sriram", "location" : "Andhra Pradesh", "sex"
 : "male" }
>
```

THEN, YOU CAN LOGIN INTO THE APP AND REACH THE MAINPAGE:

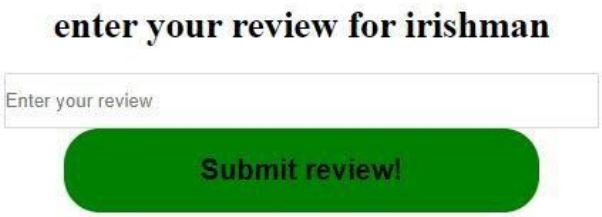
THE IRISHMAN

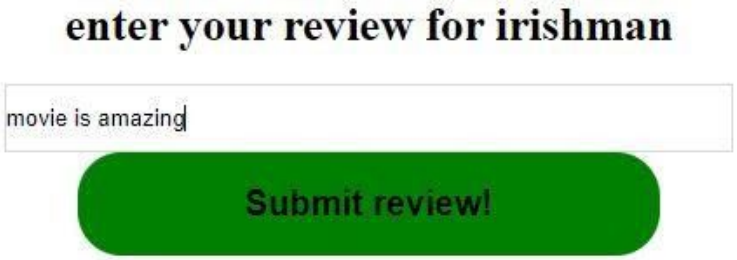
give review

see review

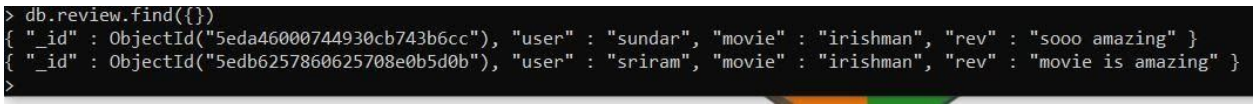
JOKER

HERE, YOU CAN GIVE A REVIEW TO THE MOVIE, WHICH REDIRECTS YOU TO:





AFTER YOU SUBMIT, THE DATA WILL BE STORED IN DATABASE



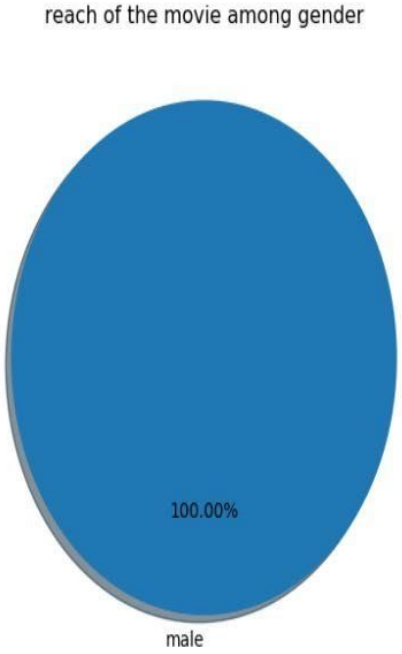
THEN WHEN YOU TRY TO GIVE REVIEW AGAIN,THE MESSAGE POPS UP AS:

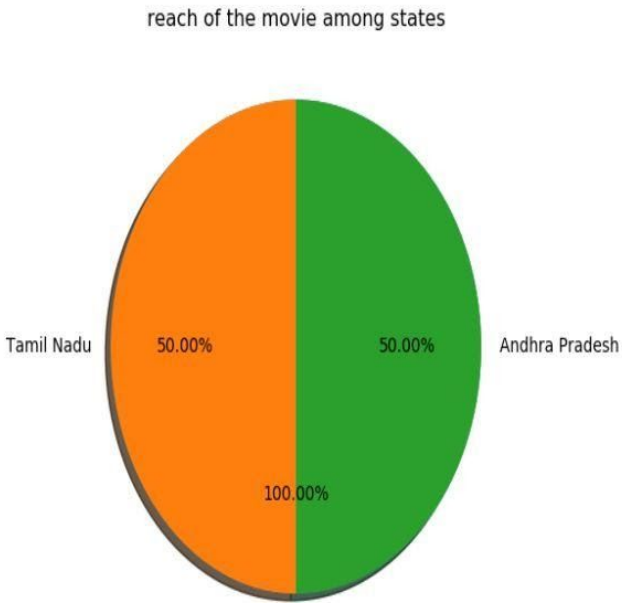
---

you already gave the review

ONCE YOU CLICK “SEE REVIEW”, YOU ARE REDIRECTED TO:

AVERAGE PUBLIC SCORE IS :79.295





THIS PAGE SHOWS THE FULL ANALYSIS OF THE IMAGE BUT LOOKING AT THE COLLECTIONS AND ANALYSING THE REVIEWS

PROJECT EXPLANATION IN DETAIL:

REQUIREMENTS: Flask, Python, Pymongo, mongodb and other libraries

The Flask app connects html page with the python file where processing of all the data is done. These data are then stored in mongo database. We created two collections namely “USERS”, and REVIEWS”. User collection stores username, password, location, sex of the user and reviews store the review, username and movie name.

At first, login page is directed where the end user logs in to the Flask app, the username and password are accepted from user and is checked with the database to authorize the user. If they don’t have an account, then they can create an account using the “sign up” page with the help of a button present in login page.

In the signup page, username, password, location and sex of the person are accepted from user and is stored in the database with the help of Flask Module which is connected with mongodb using pymongo module. After the account is created, the page automatically redirects to the login page.

Once the users login, they reach the main page where the movie name, poster along with two buttons appear, with one for giving the review and other for viewing the review. One user can provide only one review for a particular movie.

In the give page, there is also a text box where users can submit their review on the movie. Once they submit, the data will be stored in the review collection in mongo database, where it stores name, movie name and review of that respective user.

The heart of the app is “see review”, which is made different from other apps where once the user clicks the “see review” button, the page is redirected to review seeing page. In this page, all the reviews about the movie are collected, a bona fide score is given for the movie based on the reviews provided by the users on the movie. An average score is shown in the screen. Here, we also show two pie-charts showing the reach of the movie. The first pie chart shows the number of reviews given by male to female ratio. The second chart shows the reach of the movie in different states in India ,this is evaluated based on the reviews provided by the users, the user’s location is found from user collection and review from each state is found and shown in the chart.

ADVANTAGES AND LIMITATIONS:

The advantage of the project is that it helps the end user to see the trending movie in different locations and it also helps the online platforms to invest in the genre of movie in these locations. Sometimes, some movies are positively accepted by men and the movie is given better review but disliked by women so the pie chart of review, differentiating male and female will give a better understanding about the movie.

The limitation of the project is that there are some security vulnerabilities which are to be improved further and the average score of women and men are not shown. Some more analysis are needed to be done in future. The movies are added manually in the webpage by creator which can also be improved and this setting can be provided to authorize users alone.

END USERS OF THE PROJECT:

- This project is useful for people who don't believe in other online movie reviews.
- It is also useful for online movie platforms to check for better investment of movies in different locations.
- It is useful for women who believe other reviews are men based and helps to check women’s reviews into consideration before watching them.

CONCLUSION:

This app is a new attempt of showing hidden information about movie on different kind of people and their views on a film which helps us to analyze the movie much better than other models. On further development, any movie suggestion can also be added.